

Access to a collective resource

FIELD OF THE INVENTION

The invention relates to the access to a collective resource by a set of functions. The invention can be used, for example, in a data processing system comprising a plurality of processors which should access a collective memory. A example of such a system is an MPEG decoder (MPEG is an acronym for Motion Picture Expert Group).

BACKGROUND OF THE INVENTION

The access to the collective resource can be controlled by means of an interface. The interface uses an access scheme in accordance with which the functions can access the collective resource.

A possible access scheme is characterized by a repetitive access pattern. The access pattern comprises a plurality time slots. Each time slot is allocated to a given function. This means that the relevant function has access to the collective resource during this time slot. Let it be assumed that a functional system has three functions A, B and C. An access pattern may give access to the function A during two units of time, subsequently to B during one unit of time and finally to C during three units of time. This pattern thus comprises six units of time in total and is repeated every six units of time. Such an access scheme will be referred to hereinafter as a fixed access scheme.

Another possible access scheme is characterized by an order of priority. A function should submit a request to access the collective resource. The interface receives all the requests. From the functions having a pending request it allows the function having the highest priority to access the collective resource. Let it be assumed again that a functional system has three functions A, B and C. Let the order of priority be A, B, C. In this case, the function A will have access to the collective resource independently of the fact that the function B or C has a pending request. The function B will have access to the collective resource provided that there is no pending request from the function A. The function C will have access to the collective resource provided that there is no pending request from the function A or B. Such a scheme will be referred to hereinafter as a priority access scheme.

Sub
ai

The two access schemes described hereinbefore are mentioned in the international Patent Application published under the number WO95/31874. Said Application relates to an MPEG decoder.

5 SUMMARY OF THE INVENTION

It is an object of the invention to enable a cost reduction to be achieved.

The invention takes into consideration the following aspects. In principle, each function should have satisfactory access to the collective resource so as to allow said function to be performed correctly. The access to the collective resource by any function is determined
10 by two factors: firstly, the access capacity provided by the collective resource and, secondly, the access scheme that is used.

In principle, it is possible to guarantee satisfactory access for any function by giving the collective resource a comparatively high access capacity. However, in general, the collective resource will become more expensive as the access capacity provided by the
15 collective resource increases. Let it be assumed that the collective resource is, for example, a memory. The access capacity provided by the memory is then generally expressed in terms of pass band or speed. In general, a memory will be more expensive according as the speed of the memory is higher.

The fixed access scheme in fact divides the access capacity provided by the
20 collective resource among the different functions in a predetermined manner. Therefore, it suffices to ensure that the access capacity and the division of this capacity allow the collective resource to be accessed by each function in a satisfactory manner.

The access required by a function may vary in time. It is possible that a function should access the collective resource intensively for a given period and that this
25 function does not require such an intensive access for another period.

For each function, the fixed access scheme should allow for the worst-case situation in terms of access requirement. This means that if there are one or more functions whose access requirement varies considerably in time, the access capacity provided by the collective resource will be underused for most of the time. In other words, the collective
30 resource will not be used in a very efficient manner. In terms of cost, the fixed access scheme provides only a moderate return on the investment in the collective resource.

The priority access scheme gives access to the collective resource on request. This scheme thus provides flexibility and, consequently, a more efficient use of the collective resource. This means that, in principle, the priority access scheme requires a smaller access

capacity for the collective resource than the fixed access scheme. As a result of this, the priority access scheme in principle makes it possible to achieve a cost reduction for the collective resource.

However, the priority access scheme has some drawbacks. In general, it is difficult to ascertain whether each function has satisfactory access to the collective resource. Moreover, it is generally difficult to verify whether a function has always access in a critical time following a request. The verification problem is caused by the fact that the time span of access operations of a function as well as the time between a request and the resulting access depend on the behavior of the different functions as regards their requests. Therefore, intricate calculations or simulations of a statistical nature are necessary in order to verify if a given priority access scheme enables a correct operation to be achieved. This, in itself, already involves certain costs in the stage of development.

The following example illustrates a lock-up problem which may arise in the case of a priority scheme. There are three functions A, B and C in order of priority. At a given instant the function A has access to the collective resource. During this access the functions B and C submit a request. When the access by the function A has been completed, the function B will be given access to the collective resource. The request from the function C remains pending. The function A submits a request during the access of the function B to the collective resource. When the access by the function B has been completed the function A will again be given access to the collective resource. If this continues, the function A submits a request during the access by the function B and vice versa, access to the collective resource by the function C is effectively locked up. This lock-up continues until the instant at which the function A or B or both functions submit their requests at a lower frequency.

Sub 92 A lock-up as described hereinbefore can be avoided by taking care that the access operations by the functions of the highest priorities have comparatively great lengths. Thus, after having accesses the collective resource such a function will only need a new access after a comparatively long period. In a way, the long access ensures that the function is taken care of in such way that it remains silent for a comparatively long period. As a result, the frequency of the requests by the functions of the highest priorities will be comparatively low. This provides room to allow the functions of lower priorities to access the collective resource.

However, the solution to the lock-up problem described in the preceding paragraph has some disadvantages. The time between a request and the resulting access will be comparatively long for a function of low priority. Moreover, said solution requires that the

functions of the highest priority have a comparatively high storage capacity for the temporary storage of items forming the object of an access to the collective resource. These storage capacities entail a certain cost. Another solution to the lock-up problem described in the preceding paragraph will be to increase the access capacity of the collective resource.

5 However, this also entails a certain cost.

<sup>Sub
a3</sup> According to the invention, a system as defined in the opening paragraph comprises an interface adapted to implement an access scheme characterized by a plurality of states passed through in a predetermined manner, a state forming a possibility of access of a given length and defining an order of priority in accordance with which a function can access
10 the collective resource.

The invention thus guarantees a certain minimum access to the collective resource for any function having the highest priority in one or more states. Moreover, the invention guarantees for such functions a certain maximum time between a request and the resulting access. This facilitates the design of the system and also enables the functions to
15 have a comparatively moderate storage capacity. The invention consequently enables a cost reduction to be achieved in comparison with the priority scheme as described hereinbefore.

At the same time, the invention provides flexibility in the sense that a given function can access the collective resource intensively for a given period of time if required. As a consequence, the collective resource will be used more efficiently in comparison with
20 the fixed access scheme. As a result, the collective resource can have an access capacity which is reduced in comparison with that for the fixed access scheme. The invention thus provides a cost reduction in comparison with the fixed access scheme.

The invention will now be described in more detail hereinafter with reference to the drawings.

25 BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 shows a data processing system in accordance with the invention;

Figure 2 illustrates the operation of the memory interface of the system;

Figure 3 shows a data processing unit of the system;

30 Figure 4 shows the memory interface of the system;

Figure 5 illustrates a read access by a unit;

Figures 6a and 6b illustrate an arbitration for access to a collective memory;

Figure 7 shows an access interface of the memory interface;

Figure 8 shows a buffer memory circuit of the memory interface;

Figure 9 shows a read buffer memory circuit;

Figure 10 illustrates basic characteristic features of the invention as defined in Claim 1.

Figure 11 shows an arbiter for managing the access to a collective resource by
5 different processes;

Figure 12 shows a scheme for the selection of a process in the arbiter shown in Figure 11.

DESCRIPTION OF EMBODIMENTS

10 The following remarks relate to the reference signs. Similar entities bear identical reference letters in all the Figures. A plurality of similar entities may appear in a single Figure. In that case a numeral is suffixed to the reference letters in order to distinguish between similar entities. The numeral or suffix may be omitted for convenience. This applies both to the description and to the Claims.

15 Figure 1 shows a data processing system. It comprises a collective memory SDRAM, a memory interface INT, and three data processing units B1, B2 and B3. These data processing units will be referred to hereinafter as "units". Each unit B is connected to the memory interface INT via a private read bus BBR and a private write bus BBW. Each private read bus BBR and each private write bus BBW is dedicated to a given unit B. The memory
20 interface INT is connected to the collective memory SDRAM via a collective bus BM.

The general operation of the data processing system is as follows. On request the units B receive data to be processed, stored in the collective memory SDRAM. After having processed said data the units B apply the processed data to the collective memory SDRAM via the memory interface INT. The memory interface INT controls the access to the
25 collective memory SDRAM by the various units B.

The memory interface INT has two basic functions. First of all, it performs an arbitration between the various units B at the level of access to the collective memory SDRAM. A single unit B can access the collective memory SDRAM either for reading or writing at one time. This means that a unit B can access the memory only in a burst mode.
30 Secondly, in the case of reading, the memory interface INT transforms the data bursts coming from the collective memory SDRAM and intended for a given unit B into a substantially steady data stream. This data stream is thus transferred to the unit B via the respective private read bus BBR. In the case of writing the memory interface INT transforms a substantially

steady data stream coming from a given unit B into data bursts to be written into the collective memory SDRAM.

Figure 2 illustrates the operation of the memory interface INT. T(BM) represents a data transfer between the collective memory SDRAM and the memory interface INT via the collective bus BM. T(BBR1), T(BBR2) and T(BBR3) represent data transfers between the memory interface INT and the units B1, B2 and B3, respectively, via the respective private read buses BBR1, BBR2 and BBR3. T(BBW1), T(BBW2) and T(BBW3) represent data transfers between the memory interface INT and the units B1, B2 and B3, respectively, via the respective private write buses BBW1, BBW2 and BBW3.

The data transfer T(BM) consists of data bursts DB. Each data burst DB corresponds to one access operation of the collective memory SDRAM by a unit B either in the write mode or in the read mode. The parenthesized references following DB indicate to which unit B the data in the burst belong and also indicate the type of access: write (W) or read (R). For example, DB1(B1/R) indicates that the data burst DB1 concerns an access to the collective memory SDRAM in the read mode by B1.

Figure 2 shows that the memory interface INT performs a "smoothing" of the data bursts coming from the collective memory SDRAM and belonging to a certain unit B. This Figure also shows that, conversely, the memory interface INT performs a time concentration of the data received from a unit B in order to write said data into the collective memory SDRAM as a burst (data compression). Thus, the transfers of data via the private read buses BBR and the private write buses BBW are effected at comparatively low rates. Consequently, this makes it possible for the private read buses BBR and the private write buses BBW to have comparatively narrow pass bands and, as a result, to have a comparatively small width. In this respect it is to be noted that the size of a bus need not necessarily correspond to the number of bits contained in the data transferred by this bus. For example, 16 bit data can be split into 4 bit words. Thus, said data can be transferred via a 4-bit bus in the form of a sequence of 4 words.

Figure 3 shows a unit B. The unit B comprises a processor P and a general addressing circuit AGA. The processor P makes logic requests LRQ. It is assumed that the unit B processes video data, in which case a logic request LRQ may be, for example, a request for pixels of a given line in the current image. The general addressing circuit AGA transforms the logic request LRQ into a physical request PRQ. The physical request PRQ defines the physical addresses where the requested data is stored in the collective memory SDRAM. The physical requests PRQ may have the following form: a start address, a number

of addresses to be searched starting from said address and, if applicable, a scheme to be employed during the search for data. This schema may be defined in the form of: the number of consecutive addresses to be read, the number of addresses to be jumped and the number of "read and jump" iterations. The AGA may be programmable in such a manner that translation parameters define the translations of logic requests LRQ into physical requests PRQ. This enables a flexible storage of data into the collective memory SDRAM.

Figure 4 shows the memory interface INT. The memory interface INT comprises an arbiter ARB, an access interface SIF, a buffer memory circuit BUF and addressing and macro-command circuits AGB. There is an addressing and macro-command circuit AGB for each unit B.

On the whole, the internal operation of the memory interface INT is as follows. Each addressing and macro-command circuit AGB divides a physical request from the unit B with which it is associated into macro-commands. A macro-command represents a request for access to a given row in the memory. Before a macro-command is submitted to the arbiter ARB the addressing and macro-command circuit AGB verifies whether there is enough room in the buffer memory circuit BUF. To this end, it first applies the macro-command to the buffer memory circuit BUF. If the buffer memory circuit BUF confirms that there is room for the storage of the number of data defined by the macro-command, the addressing and macro-command circuit AGB submits the macro-command to the arbiter ARB. The arbiter ARB receives the macro-commands from the various addressing and macro-command circuits AGB and selects a macro-command to be applied to the access interface SIF. This selection is made in accordance with an arbitration scheme described hereinafter. The access interface SIF processes the macro-commands received from the arbiter ARB in the order of their receipt. Thus, the access interface SIF provides the access to the collective memory SDRAM, the access operations being defined by the macro-command being processed.

A macro-command enables X groups of addresses to be accessed, each group comprising Y addresses, the groups of addresses being separated from one another by Z words, X, Y and Z being integers. Thus, a macro-command contains the following information:

- first address to be accessed;
- number of addresses to be accessed following upon the first address in a group of addresses (Y-1);
- number of addresses to be skipped between two groups of consecutive addresses (Z);

- number of address groups to be accessed in addition to the first group (X-1);
- type of access: read or write.

An example of a macro-command at bit level is as follows. It is assumed that the data stored in the collective memory SDRAM has a width of 32 bits and the collective memory SDRAM has a maximum capacity of 256 Megabit. This means that an address is expressed as 23 bits. It is further assumed that the access operations are limited to a maximum size of 16 addresses. Such a limit is preferred in view of latency. Thus, X-1 and Y-1 are 15 at the most and can consequently be coded in 4 bits. Finally, a row contains a maximum of 512 addresses in accordance with the configuration of the collective memory SDRAM. Consequently, the number of addresses to be jumped cannot exceed 511 and, as a result, this number can be coded in 9 bits. The macro-commands thus have a size of $23+2 \times 4+9+1 = 41$ bits. The address can be coded in the bits 40 to 18, the access type in the bit 17, the number of words to be read (Y-1) in the bits 16 to 13, the number of words to be jumped (Z) in the bits 12 to 4, and the number of word groups (X-1) in the bits 3 to 0.

Figure 5 illustrates a process of access to the collective memory SDRAM in the read mode by a given unit B. The horizontal dimension represents the time. The vertical dimension of this diagram represents the different functional elements which are involved. The diagram contains arrows. These arrows represents various steps S in the process of access to the interface memory SRAM.

S1 = The processor P of the relevant unit B submits a logic request LRQ to the general addressing circuit AGA. The logic request LRQ specifies a data sub-set, for example the luminance pixels of a line in a set of data to be processed, for example an image.

S2 = The general addressing circuit AGA transforms the logic request LRQ into a physical request PRQ.

S3 = The general addressing circuit AGA submits the physical request PRQ to the macro-commands addressing circuit AGB.

S4 = The macro-command addressing circuit AGB transforms the physical request PRQ into macro-commands.

S5 = The macro-command addressing circuit AGB submits the first one of the macro-commands derived from the physical request PRQ to the buffer memory circuit BUF.

S6 = The buffer memory circuit BUF verifies whether there is room for the storage of the number of data specified by the macro-command.

S7 = The buffer memory circuit BUF sends an acknowledgement to the macro-command addressing circuit AGB that there is room.

S8 = Represents a given delay.

S9 = The macro-command addressing circuit AGB submits the macro-command to the arbiter ARB.

5 S10 = The arbiter ARB processes the macro-command as a request for access to the collective memory SDRAM in accordance with an arbitration schema which applies to any access to the collective memory SDRAM by the units (in the read and in the write mode).

S11 = The arbiter ARB submits the macro-command to the access interface SIF.

10 S11a = The arbiter ARB sends an acknowledgement to the buffer memory circuit BUF that the macro-command has been submitted to the access interface SIF.

S12 = The macro-command is queued in the access interface SIF, which first processes the previously received macro-commands.

15 S13 = The access interface SIF generates control signals for the collective memory SDRAM on the basis of the macro-command. These control signals results in the data at the addresses specified by the macro-command are read successively.

S14 = The data successively read from the collective memory SDRAM is transferred to the buffer memory circuit BUF.

S15 = The buffer memory circuit BUF temporarily stores the data.

20 S16 = The buffer memory circuit BUF transfers the data to the processor P in a substantially steady manner.

The steps S5 - S15 are repeated for each macro-command following the logic request LRQ made in the step S1.

25 The next steps are not shown in Figure 5. Upon the step S1 the general addressing circuit AGA sends an acknowledge signal to the processor P, which signal indicates that the logic request LRQ has been accepted and will be processed. In response to this acknowledge signal the processor P makes a new logic request and holds it till further notice. When the macro-command addressing circuit AGB submits the last macro-command in response to the logic request LRQ the processing of the logic request LRQ is finished. In that case the macro-command addressing circuit AGB sends an acknowledge signal to the
30 general addressing circuit AGA to indicate to the latter that the processing of the logic request LRQ has been completed. In response thereto, the general addressing circuit AGA starts the processing of the new logic request LRQ in a manner similar to the processing of the logic request LRQ effected in the step S1. In other words, the process is repeated.

Figures 6a and 6b show an arbitration scheme for the arbiter ARB. In Figure 6a eight states ST1-ST8 are shown as circles. These states ST occur one after the other and in a cyclic fashion. Each state ST represents a possibility of sending a macro-command to the access interface SIF. Thus, each state represents a possibility of accessing a memory. Each state corresponds to a given processor P. The processor P to which a given state corresponds is specified in the circle representing the state.

Figure 6b represents the arbitration process corresponding to Figure 6a. This process comprises a plurality of steps SA1-SA4 and is carried out for each state ST in Figure 6a. The step SA1 is the first step carried out after a jump of state. In the step SA1 the arbiter ARB verifies if a macro-command, submitted by the macro-command addressing circuit AGB and following a logic request LRQ from the processor P[j] to which the state S[i] corresponds, is pending. If such a macro-command is pending the step SA1 is followed by the step SA2. In the step SA2 the arbiter ARB sends the relevant macro-command to the access interface SIF. The effect of this is that after a given delay the collective memory SDRAM will be accessed by the relevant processor P as defined by the macro-command. After the macro-command has been sent the arbiter jumps to the next state, which means that the process represented in Figure 6b is repeated. However, if in the step S1 the arbiter ARB has detected that there is no pending macro-command associated with the processor P to which the state S[i] corresponds, the step SA1 is followed by the step SA3. In the step SA3 the arbiter ARB verifies if other macro-commands are waiting. If there are no other macro-command waiting the arbiter ARB jumps to the next state and the process shown in Figure 6b is repeated. If there are other macro-commands waiting the arbiter ARB performs the step SA4. In the step SA4 the arbiter ARB selects a macro-command in an order of priority. Each macro-command has a given level of priority. The priority level is determined by the processor P from which the macro-command originates. The arbiter ARB thus selects the macro-command having the highest priority level and sends this macro-command to the access interface SIF. After having sent the macro-command the ARB jumps to the next state, which means that the process shown in Figure 6b is repeated.

With regard to Figures 6a and 6b it is to be noted that it is not necessary that each state corresponds to a processor P. It is possible to introduce one or more states which do not correspond to any processor P, which means that it is possible to introduce free states. In the case of a free state the arbiter ARB selects a macro-command only on the basis of the priority order. Such a free state can be useful in the case that the data processing system includes a processor P whose restraints in terms of latency time and pass band at the level of

access to the collective memory SDRAM are comparatively moderate. Thus, in order to avoid that this processor P is not given enough access, free states could be introduced. The processor P can take advantage of these free states to access the collective memory SDRAM.

Figure 7 shows an example of the access interface SIF. The access interface SIF comprises a macro-command buffer memory FIFO_MC, a column generator CAGU, a command generator CGU, a control signal generator IF_SDRAM, a data buffer memory IF_D.

The general operation of the access interface SIF is as follows. The macro-command buffer memory FIFO_MC receives the macro-commands from the arbiter ARB. This memory temporarily stores these macro-commands and sends them to the column generator CAGU in the order of their arrival. In the case that the macro-command buffer memory FIFO_MC is full and therefore cannot accept a new macro-command, it reports this to the arbiter ARB. As a result of this signal from the access interface SIF, which indicates that its "FIFO is full", the arbiter ARB postpones the transfer of the currently selected macro-command until the instant at which the macro-command buffer memory FIFO_MC reports that it can accept a new macro-command. In fact, the "FIFO is full" signal from the access interface SIF "freezes" the arbiter ARB for a certain time.

The column generator CAGU requests a new macro-command from the macro-command buffer memory FIFO_MC when the memory access operations in accordance with the preceding macro-command have been carried out. As a matter of fact, the column generator CAGU in combination with the command generator CGU translate the macro-command into a series of addresses. An address of the collective memory SDRAM is defined by the number of a bank of the collective memory SDRAM, the number of a row and the number of a column. It has already been stated that a macro-command relates to an access to a single row of the interface memory SRAM, which automatically implies that the access takes place in a single bank. Therefore, it suffices that the column generator CAGU generates a series of columns on the basis of the macro-command in order to define a series of addresses in accordance with the macro-command. An implementation of the column generator CAGU may comprise, for example, some counters and some logic circuits. In such an implementation the content of a macro-command serves to program the counters.

The command generator CGU successively receives another column number of the collective memory SDRAM. The command generator CGU further receives from the macro-command buffer memory FIFO_MC the bank number and the row number of the addresses as defined by the macro-command. This information enables the column generator

CAGU to define a series of commands for access to the collective memory SDRAM, each command defining a single address. Moreover, the command generator CGU generates commands necessary for setting the collective memory SDRAM to a correct state so as to allow the access operations as defined by the macro-commands. These commands relate to the appropriate processes for the collective memory SDRAM, such as preloading and activation. Furthermore, the command generator CGU ensures that the collective memory SDRAM is refreshed at regular intervals and generates the commands necessary to perform these refresh operations.

The control signal generator IF_SDRAM generates control signals on the basis of the commands received from the command generator CGU. For example, the control signal generator IF_SDRAM generates signals referred to by the acronyms RAS, CAS. The control signal generator IF_SDRAM ensures that in a series of control signals certain waiting times for the collective memory SDRAM are observed. These waiting times may vary depending on the type of collective memory SDRAM that is used. Consequently, the control signal generator IF_SDRAM is specific to the type of collective memory SDRAM that is used. If a collective memory SDRAM of another type is to be used it is sufficient to modify, i.e. reprogram, the control signal generator IF_SDRAM. In principle, the other elements of the access interface do not require any modifications.

The data buffer memory IF_D serves to transfer the data from the collective memory SDRAM to the buffer memory shown in Figure 4 in the case of reading and to transfer the data from the buffer memory BUF to the collective memory SDRAM in the case of writing. For this purpose, the data buffer memory data buffer memory IF_D synchronizes the data supplied by the collective memory SDRAM (reading) or applied to the collective memory SDRAM (writing). Moreover, the data buffer memory IF_D forms a FIFO having a unit depth. This means that if a given clock pulse causes data to be read from the collective memory SDRAM this data will be transferred to the buffer memory circuit BUF on the next clock pulse. In the case of writing this is the other way round.

Figure 8 shows an example of the buffer memory circuit BUF which forms part of the memory interface INT shown in Figure 4. The buffer memory circuit BUF comprises a read buffer memory circuit BUFR as well as a write buffer memory circuit BUFW and a buffer memory for acknowledge signals FIFO_ACK. The read buffer memory circuit BUFR and the write buffer memory circuit BUFW are connected to the collective memory SDRAM via the access interface SIF and the collective bus BM as is shown in Figure 1. The read buffer memory circuit BUFR is connected to the units B1, B2 and B3 via

the private read buses BBR1, BBR2 and BBR3, respectively. The write buffer memory circuit BUFW is connected to the units B1, B3 and B3 via the private write buses BBW1, BBW2 and BBW3, respectively. The acknowledge signal buffer memory FIFO_ACK is connected to the arbiter ARB.

5 The general operation of the buffer memory circuit BUF is as follows. The read buffer memory circuit BUFR temporarily stores the data received from the collective memory SDRAM, while the write buffer memory circuit BUFW stores the data to be written into the collective memory SDRAM and received from the different units B. The acknowledge signal buffer memory FIFO_ACK receives the acknowledge signals coming
10 from the arbiter ARB. Such a signal indicates that the arbiter ARB has applied a macro-command to the access interface SIF.

 The acknowledge signal buffer memory FIFO_ACK has the same depth as the macro-command buffer memory FIFO_MC of the access interface SIF shown in Figure 7. Consequently, when a macro-command leaves the macro-command buffer memory
15 FIFO_MC, as a result of which a memory access in accordance with the macro-command is effected, the acknowledge signal corresponding to this macro-command leaves the acknowledge signal buffer memory FIFO_ACK. This signal indicates whether the relevant access operation is a read access or a write access. In the first-mentioned case the read buffer memory circuit BUFR is activated to receive data from the collective memory SDRAM,
20 while in the last-mentioned case the write buffer memory circuit BUFW is activated to send data to the collective memory SDRAM. The acknowledge signal supplied by the acknowledge signal buffer memory FIFO_ACK further indicates the number of data involved in the access operation as defined by the macro-command. This indication is used by the buffer memory circuit BUF for the internal management of "where to store the data or where
25 to fetch the data?" in the case of reading or writing, respectively.

 Figure 9 shows an example of the read buffer memory circuit BUFR. The read buffer memory circuit BUFR comprises an input buffer memory IB, an interface memory SRAM, an arrangement comprising a plurality of output buffer memories OB, an arrangement of control circuits CON, and an interface memory access arbiter ARBBR. The input buffer
30 memory IB is connected to the collective memory SDRAM via the access interface SIF already shown in Figure 4. The output buffer memories OB1, OB2 and OB3 are connected to processors P1, P2 and P3 via the private read buses BBR1, BBR2 and BBR3, respectively, the last-mentioned elements being shown in Figures 2 and 3. The control circuits CON1, CON2 and CON3 are connected to the macro-command addressing circuit AGB1, the macro-

command addressing circuit AGB2 and the macro-command addressing circuit AGB3, respectively, as well as to the access interface SIF.

The read buffer memory circuit BUFR operates as follows. The data received from the collective memory SDRAM has a width of N bits, N being an integer, and arrive with a frequency F . The interface memory SRAM has a width of $2N$ bits, which means that an address can contain $2N$ bits, and operates at the frequency F . The input buffer memory IB forms pairs of two consecutive data coming from the collective memory SDRAM and loads these pairs into the interface memory SRAM. It takes two clock cycles to form one pair. When it is assumed that all the successive data received from the collective memory SDRAM can be paired, a write access to the interface memory SRAM will be effected every two clock cycles. Access to a single address requires only one clock cycle. Thus, between two write access operations one clock cycle is available to access the interface memory SRAM for reading, in order to transfer the data read from the collective memory SDRAM to the units B. Thus, in principle, write access and read access to the interface memory SRAM can be effected alternately and one after another. The access to the interface memory SRAM will be described in greater detail hereinafter.

The interface memory SRAM is, in fact, divided into three zones $Z1$, $Z2$ and $Z3$. The zones $Z1$, $Z2$ and $Z3$ contain the data intended for the processors $P1$, $P2$ and $P3$, respectively. The data from the collective memory SDRAM is written into the zone $Z1$, $Z2$ or $Z3$ via the IB depending on the processor P from which the current macro-command originates. The data present in the zones $Z1$, $Z2$ and $Z3$ is transferred to the output buffer memories $OB1$, $OB2$ and $OB3$, respectively, in a substantially steady fashion and in accordance with a substantially fixed scheme. An output buffer memory OB in fact splits the data into a plurality of parts and applies the relevant data to the respective processor P part by part. For example, an output buffer memory OB can split 16 bit data into 4-bit parts. Thus, instead of transferring the data in a single clock cycle, which requires a 16-bit bus, the data is transferred part by part in 4 clock cycles, which requires only a 4-bit bus.

The control circuits $CON1$, $CON2$ and $CON3$ control the zones $Z1$, $Z2$ and $Z3$, respectively. For this purpose, each control circuit CON controls a group of parameters. These parameters include a write pointer, a read pointer and a zone occupancy value. The write pointer defines the address in which the data from the collective memory SDRAM is to be written. The read pointer defines the address of the data to be transferred to the relevant output buffer memory OB . The occupancy value indicates the number of addresses still available for the storage of the data received from the collective memory SDRAM. The

control circuits CON1, CON2 and CON3 also control the output buffer memories OB1, OB2 and OB3, respectively. For this purpose, each control circuit CON controls a parameter representing the state of occupancy of the corresponding output buffer memory OB.

The control process carried out by a control circuit CON will now be described, assuming that the collective memory SDRAM is accessed as illustrated in Figure 5. In the step S5 the macro-command addressing circuit AGB submits a macro-command to the buffer memory circuit BUF. This macro-command is processed by the control circuit CON corresponding to the processor P from which the macro-command originates. The control circuit CON compares the number of data defined by the macro-command with the occupancy value. Thus, the control circuit CON verifies whether there is sufficient room in the relevant zone Z to store the desired data. If there is enough room, the control circuit CON reports this to the macro-command addressing circuit AGB and, moreover, updates the occupancy parameter. This means that it considers the data as already stored in the relevant zone while this should still be done. Therefore, the update of the occupancy parameter may be regarded as a reservation in the relevant zone.

What happens during the step S12 as shown in Figure 5 will now be described. This step S12 represents a read-out of the collective memory SDRAM in accordance with the relevant macro-command. It has already been explained that at the instant at which the access interface SIF starts to process the macro-command and thus starts the read-out, the acknowledge signal associated with the relevant macro-command leaves the acknowledge signal buffer memory FIFO_ACK shown in Figure 7. This acknowledge signal indicates that the access is a write operation and, moreover, this signal specifies the processor P from which the macro-command stems. Thus, the control circuit CON corresponding to this processor P knows that it should supply the addresses where the data should be stored in the collective memory SDRAM. Furthermore, the control circuit CON receives an indication of the number of data involved in the access operation in accordance with the macro-command, which indication forms part of the acknowledge signal.

Each time that a data pair from the collective memory SDRAM is written into the relevant zone Z the control circuit CON increments the write pointer. Moreover, it updates the occupancy value. The control circuit CON continues to do so until the read access to the collective memory SDRAM as defined by the macro-command has been completed. The control circuit CON detects the end of the access operation with the aid of the number of data involved in the access operation, this number being indicated by the acknowledge signal, and a by counting the data written into the interface memory SRAM.

After each read-out of a data pair from a given zone Z the control circuit CON which controls this zone increments the read pointer. Moreover, it updates the occupancy value.

The interface memory access arbiter ARBBR controls the access to the interface memory SRAM. There are different types of access: (1) access by the access interface SIF to write data from the collective memory SDRAM into the interface memory SRAM, (2) access operations by the output buffer memory OB1, (3) access operations by the output buffer memory OB2, and (4) access operations by the output buffer memory OB3. The three last-mentioned access operations serve to transfer data contained in the interface memory SRAM to the processors P1, P2 and P3, respectively.

Any access to the interface memory SRAM is effected in response to a request submitted to the interface memory access arbiter ARBBR. From the current requests the interface memory access arbiter ARBBR selects the request having the highest priority. The write access requests (access via the access interface SIF) having the maximal priority. As a result of the fact that pairs of data are written, as explained hereinbefore, such a request generally occurs every two clock cycles. A write operation requires only one clock cycle. Consequently, there will be enough opportunities to access the interface memory SRAM in the read mode in order to transfer the data to the various processors P.

The read access requests by a given output buffer memory OB are made depending on the size of the private read bus BBR between the output buffer memory OB and the unit B. It is assumed, for example, that the size of the bus is $N/2$ bits. This means that a $N/2$ bit part can be transferred from the output buffer memory OB to the unit B in every clock cycle. Reading of the interface memory SRAM is effected in pairs of data. One data pair comprises $2N$ bits. Therefore, 4 clock cycles are needed to send one data pair to the unit B. The transfer of one data pair involves a request for access to the interface memory SRAM in the read mode. Thus, in the present example, the output buffer memory OB will submit an access request every 4 clock cycles. This example shows that the sizes of the buses to the units B dictate the frequency of the access requests of the various output buffer memories OB. If the size of the private read bus BBR is $N/4$ bit there will be an access request every 8 clock cycles.

The following description relates to an example of arbitration for access to the interface memory SRAM. It is assumed that the size of the private read bus BBR1 is $N/2$ bits and that the size of the private read bus BBR2 and that of the private read bus BBR3 is $N/4$ bits. The access operations of the access interface SIF have the highest priority, followed, in

order of priority, by the access operations of the output buffer memories OB1, OB2 and OB3. Finally, it is assumed that the requests for all the types of access (SIF, OB1, OB2, OB3) are submitted simultaneously in the first clock cycle.

5 Cycle 1: all the requests are made at the same time; pending requests: the access interface SIF, the output buffer memories OB1, OB2 and OB3.

Cycle 2: the interface access SIF having the highest priority has the lead and drops its request; the output buffer memories OB1, OB2 and OB3 maintain their requests; pending requests: the output buffer memories OB1, OB2 and OB3.

10 Cycle 3: the output buffer memory OB1, which has the second highest priority, has the lead and drops its request; the access interface SIF submits a new request; pending requests: access interface SIF, the output buffer memories OB2 and OB3.

Cycle 4: the access interface SIF having the highest priority has the lead and drops its request; the output buffer memories OB2 and OB3 maintain their requests; pending requests: the output buffer memories OB2 and OB3.

15 Cycle 5: the output buffer memory OB2, which has the third highest priority, has the lead and drops its request; the access interface SIF submits a new request; pending requests: the access interface SIF and the output buffer memory OB3.

20 Cycle 6: the access interface SIF having the highest priority has the lead and drops its request; the output buffer memory OB1 is running out of buffer capacity and again submits a request; pending requests: the access interface SIF, the output buffer memories OB1 and OB3.

Cycle 7: the output buffer memory OB1, which has the second highest priority, has the lead and drops its request; the access interface SIF submits a new request; pending requests: the access interface SIF, the output buffer memory OB3.

25 Cycle 8: the access interface SIF having the highest priority has the lead and drops its request; the output buffer memory OB3 maintains its request; pending request: output buffer memory OB3.

30 Cycle 9: the output buffer memory OB3, which has the fourth highest priority, has the lead and drops its request; the access interface SIF submits a new request; pending request: the access interface SIF.

Cycle 10: the access interface SIF having the highest priority has the lead and drops its request; the output buffer memory OB1 is running out of buffer capacity and again submits a request; pending request: the output buffer memory OB1.

Cycle 11: the output buffer memory OB1, which has the second highest priority, has the lead and drops its request; the access interface SIF submits a new request; pending request: the access interface SIF.

5 Cycle 12: the access interface SIF having the highest priority has the lead and drops its request; the output buffer memory OB2 is running out of buffer capacity and again submits a request; pending request: the output buffer memory OB2.

Cycle 13: the output buffer memory OB2 having the third highest priority has the lead and drops its request; the access interface SIF submits a new request; pending request: the access interface SIF.

10 Cycle 14: the access interface SIF having the highest priority has the lead and drops its request; the output buffer memory OB1 is running out of buffer capacity and again submits a request; pending request: the output buffer memory OB1.

15 Cycle 15: the output buffer memory OB1 having the second highest priority has the lead and drops its request; the access interface SIF submits a new request; pending request: the access interface SIF.

Cycle 16: the access interface SIF having the highest priority has the lead and drops its request; the output buffer memory OB3 is running out of buffer capacity and again submits a request; pending request: the output buffer memory OB3.

20 Cycle 17: the output buffer memory OB3 having the fourth highest priority has the lead and drops its request; the access interface SIF submits a new request; pending request: the access interface SIF.

Cycle 18: the access interface SIF having the highest priority has the lead and drops its request; the output buffer memory OB1 is running out of buffer capacity and again submits a request; pending request: the output buffer memory OB1.

25 Cycle 19: the output buffer memory OB1 having the second highest priority has the lead and drops its request; the access interface SIF submits a new request; pending request: the access interface SIF.

30 Cycle 20: the access interface SIF having the highest priority has the lead and drops its request; the output buffer memory OB2 is running out of buffer capacity and again submits a request; pending request: the output buffer memory OB2.

Cycle 21: the output buffer memory OB2 having the third highest priority has the lead and drops its request; the access interface SIF submits a new request; pending request: the access interface SIF.

Cycle 22: the access interface SIF having the highest priority has the lead and drops its request; the output buffer memory OB1 is running out of buffer capacity and again submits a request; pending request: the output buffer memory OB1.

5 Cycle 23: the output buffer memory OB1 having the second highest priority has the lead and drops its request; the access interface SIF submits a new request; pending request: the access interface SIF.

Cycle 24: the access interface SIF having the highest priority has the lead and drops its request; the output buffer memory OB3 is running out of buffer capacity and again submits a request; pending request: the output buffer memory OB3.

10 Cycle 25: the output buffer memory OB3 having the fourth highest priority has the lead and drops its request; the access interface SIF submits a new request; pending request: the access interface SIF.

Cycle 26: the access interface SIF having the highest priority has the lead and drops its request; the output buffer memory OB1 is running out of buffer capacity and again
15 submits a request; pending request: the output buffer memory OB1;

Etc.

The access operations described hereinbefore have a periodicity of 8 cycles. It is as though arbitration is effected by a cyclic machine having eight states. This is owing to the fact that in the example it is assumed that all the units B process their data in a regular
20 fashion.

Moreover, it is assumed that the access requests by the access interface SIF are submitted regularly once every two clock cycles. In practice, these assumptions are not necessarily correct. For this reason, it is preferred to control the access to the interface memory SRAM with the aid of an arbiter and in an order of priority rather than with the aid of a cyclic
25 machine. The arbiter allows a certain degree of flexibility as regards the access to the interface memory SRAM and, consequently, it allows a better utilization of the bandwidth available for the data transfer.

The following is an example of an exception to the rule in accordance with which there is only one write access to the interface memory SRAM (= access by the access
30 interface SIF) every two clock cycles. This exception occurs in the case of an access to the collective memory SDRAM in response to a macro-command involving an odd number of data. All the data involved in this access, except for the last data element, have a partner and thus form pairs to be written into the interface memory SRAM. The last data element is alone. It is not possible to use the next data element to form a pair because this data element

is the first data element of another access operation and is therefore intended for another processor P. As a result, this first data element of another access operation should be stored in another zone Z of the interface memory SRAM. Thus, upon the arrival in the IB of the last data element of the access involving an odd number of data elements said data is loaded into the interface memory SRAM without a partner in the clock cycle that follows the write operation for the last data pair comprised in the access operation. Consequently, there will be two write access operations in succession without a break of one clock cycle, which would otherwise allow a read access operation between two write access operations.

Figures 5 and 9 relate to the operation of the memory interface INT in the read mode. The operation in the write mode is substantially symmetrical. This means that the write buffer memory circuit BUFW is similar to the read buffer memory circuit BUFR described hereinbefore. Thus, the write buffer memory circuit BUFW comprises an interface memory divided into zones, each zone being associated with a different unit B. The units B can supply data to be written into the collective memory before or after the macro-command which indicates where this data should be stored. In fact, a zone will be filled as soon as the unit B to which the zone corresponds supplies data to be stored in the collective memory. Said filling may continue, for example, until the zone is completely filled with data. In this case the memory interface INT will indicate to the relevant unit B that there is no longer any room for the storage of new data. This will prevent the unit B from supplying new data until room has become available in the relevant zone. A data transfer from the zone Z to the collective memory results in room being made available. Such a dump can occur as soon as a macro-command is accepted by the arbiter ARB and is processed by the access interface SIF. It is likewise possible that a macro-command is issued before the relevant unit B has supplied the data. In any case, no macro-command will be applied to the arbiter ARB as long as the write buffer memory circuit BUFW does not indicate an adequate filling level of the relevant zone.

Sub A4
Figure 10 illustrates basic characteristic features of the invention as outlined in the section "Summary of the invention". A functional system comprises a set of functions F which should access a collective resource RSRC. The system comprises an interface INT adapted to implement an access scheme AS characterized by a plurality of states S which are passed through in a predetermined manner, a state S forming a possibility of access of a given length and defining an order of priority in accordance with which a function F can access the collective resource RSRC.

The data processing system illustrated in Figure 1 is an example of an implementation of the basic characteristic features illustrated in Figure 10. The collective resource RSRC in Figure 10 takes the form of the collective memory SDRAM in Figure 1. The functions F in Figure 10 take the form of the data processing units B in Figure 1. The interface INT of the data processing system illustrated in Figure 1 implements an access scheme that is illustrated in Figures 6a and 6b. The access scheme is characterized by a plurality of states ST shown in Figure 6a. Each state ST forms a possibility of access by means of a macro command. The access operation defined by a macro-command has size (length) of 16 addresses. Each state ST illustrated in Figure 6a defines an order of priority in accordance with which a data processing unit can access the collective memory SDRAM. The data processing unit whose reference sign appears in a circle representing a state ST has the highest priority for that state ST.

What follows is an example of how the invention can be applied in an integrated circuit comprising an MPEG decoder and other circuits for the processing and display of images. In such an integrated circuit various processes take place, a process being equivalent to a function. The following is an example of a process P which should access a collective memory of the SDRAM type, referred to as SDRAM hereinafter:

- P1: DISP (Display)
- P2: OSD (On Screen Display)
- P3: BCKGND (Background)
- P4: CCIR656
- P5: VMIX (Video Mixer)
- P6: VFW (Video FIFO Write)
- P7: VFR (Video FIFO Read)
- P8: AUDIOFIFO (Audio FIFO Read and Write)
- P9: AUDIOBUF (Audio Buffer Read and Write)
- P10: MC (Motion Compensation)
- P11: DMU
- P12: INTERP

The process P1 (DISP) concerns the display of the decoded images. The process P2 (OSD) concerns the display of graphical data superposed on a displayed decoded image. The process P3 (BCKGND) concerns the display of a stationary image as a background or showing through a decoded and processed image. The process P4 (CCIR656) concerns an input of the integrated circuit via which images which have already been

decoded can be received and stored in a memory. These images can be displayed instead of and at the location of the images from the MPEG decoder by means of the process P1 (DISP). The process P5 (VMIX) concerns the mixing of three data streams resulting from the processes P1 (DISP), P2 (OSD) and P3 (BCKGND). The processes P6 (VFW), P7 (VFR) and P8 (AUDIOFIFO) concern the writing and reading of encoded audio and video data extracted from a stream of MPEG data by means of a descrambling/demultiplexing circuit. The process P9 (AUDIOBUF) concerns the writing and reading of intermediate results from an audio processor. The process P10 (MC) concerns the motion compensation, which is a step in MPEG video decoding. The process P11 (DMU) concerns an acceleration circuit which serves to carry out comparatively complex operations on data in the SDRAM, such as for example a displacement of one or more data blocks or a filtering operation. The process P12 (INTERP) concerns the last part of MPEG decoding, providing decoded images to be loaded into the SDRAM in order to display them or use them as reference images, or both.

Figure 11 shows an arbiter for the management of the access to the SDRAM by the various processes P described hereinbefore. The arbiter receives the access requests from all the processes and selects one of them. A selected request enables the process which has submitted this request either to write or read 32-bit words into or out of the SDRAM. After this access to the SDRAM, the arbiter selects another request etc.

The arbiter comprises a process selector MPS and a fixed priority selector FPS having four inputs 1-4. The process selector MPS receives the requests for access to the SDRAM from the processes P1-P9. From these processes it selects one process P. This selection is made in accordance with a scheme which will be described in greater detail hereinafter. If the selected process P has submitted a request, this request will be transferred to the input 1 of the fixed priority selector FPS. If no request has been submitted no request will be transferred to this input. A request from the process P10 (MC) is transferred to the input 2 of the fixed priority selector FPS. A request from the process P11 (INTERP) is transferred to the input 3. A request from the process P12 (DMU) is transferred to the input 4.

The fixed priority selector FPS selects from the requests at the inputs 1-4 the request, if any, at the input having the lowest number. Thus, if the process P selected by the process selector MSP has submitted a request, this request will be selected. If there is no such request, the fixed priority selector FPS will select a request from the process P10 (MC). If there is no such request, it will select a request from the process P11 (INTERP). If there is no such request, the fixed priority selector PPS will select a request from the process P12

(DMU). If there is no request at the inputs 1-4 of the fixed priority selector FPS, the process selector MPS selects the next process appearing in its scheme.

Figure 12 shows a scheme in accordance with which the process selector MPS can select a process. This scheme can be implemented by means of a state machine. In this case, each circle in Figure 12 represents a state. A process P shown in a circle could be said to own the state represented by this circle. A state forms a possibility of either writing or reading eight 32-bit words into or out of the SDRAM. This possibility is reserved for the process owning the state and for the processes P10-P12 in accordance with a priority order described hereinbefore. There is one circle in which no process P is shown, an "empty" circle. The state represented by this circle could be referred to as a free state; there is no owner. In this state the access to the SDRAM is reserved exclusively for the processes P10-P12.

Indeed, each circle, or state, defines a sub-set of processes which can access the SDRAM and an order of priority among these processes. The sub-set comprises the process shown in the circle, if any, and the processes P10-P12. If one process of the sub-set has submitted a request, the access will be granted to this process. If a plurality of processes of the sub-set have submitted a request, the access will be granted to the process having the highest priority. When the access has been completed, the arbiter jumps to the next state. If no process of the sub-set has submitted a request, the arbiter will jump directly to the next state.

The scheme shown in Figure 12 forms a circle of 64 states. In this respect the following special feature of the respective states belonging to the process P8 (AUDIOFIFO) and P9 (AUDIOBUF) is to be noted. The access operations by these processes alternate between writing and reading from one cycle to the other. This means that if the process P8, or the process P9, has performed a read access in a given cycle, the process will perform a write access in the next cycle, and vice versa.

The above Figures and their description illustrate rather than limit the invention. It is evident that there are numerous alternatives within the scope of the appended Claims. In conclusion, some remarks are made in this respect.

The invention may be applied in any type of functional system. Figure 1 merely shows one possibility where the invention is applied in a data processing system. It is also possible to apply the invention in, for example, a communication system comprising a central server and a plurality of user terminals. The invention may be applied to manage the access of the user terminals to the central server in an advantageous manner.

5

10